AD-761 965

HALF-TONE PERSPECTIVE DRAWINGS BY COMPUTER

Chris Wylie, et al

Utah University

Prepared for:

Advanced Research Projects Agency

12 February 1968

Technical Report 4-2

Chris Wylie
Gordon Romney
David C. Evans
Alan Erdahl

HALF-TONE PERSPECTIVE DRAWINGS BY COMPUTER

November 14, 1967
Revised February 12, 1968
in accordance with
ARPA Technical Report 4-3.

COMPUTER SCIENCE
Information Processing Systems
University of Utah
Salt Lake City, Utah

Advanced Research Projects Agency • Department of Defense • ARPA order 829
Program code number 6D30

ABSTRACT

This paper is a brief description of an algorithm for the creation of two-dimensional, half-tone pictures of perspective projections of three-dimensional objects. Only the visible surfaces are displayed; all hidden surfaces are erased. This process is independent of the orientation of the object. The inclusion of half-tone shading was considered important because the illumination of an object gives a viewer much information about the three-dimensionality of the object. A FORTRAN IV program is working on a Univac 1108. Preliminary results indicate that this approach is not only possible, but practical for complex objects. Processing time is small and storage requirements are very compact.

HALF-TONE PERSPECTIVE DRAWINGS BY COMPUTER

By Chris Wylie
Gordon Romney
David Evans
Alan Erdahl

University of Utah
Salt Lake City, Utah

INTRODUCTION

In recent years, the sheer increase in demand for the graphic presentation of three-dimensional objects has almost overwhelmed conventional facilities; that is, designers, draftsmen and, especially, engineering artists. For example, it is important for a designer or architect to quickly describe a three-dimensional object and view it immediately; not as an endless set of engineering drawings, but as if he were viewing the three-dimensional object itself. He should be able to take a distant look at a complicated object, and then view, in detail, any subsection of the object. In other words, he would like to quickly and cheaply simulate and view the thing he is designing.

The goal of this project is to provide a system which will display images that a person can "feel", as contrasted with images that he must laboriously interpret (e.g. the engineering drawings of an airplane).

Several subjective factors apparently help the viewer's ability to "feel" the overall structure of a three-dimensional object: 1) binocular (or stereo) vision, 2) elimination of the hidden surfaces, 3) recognition of distance and shape as a function of illumination (or shading), and 4) real-time movement.

For a display algorithm to be practical, the computing time should grow only linearly with the complexity of the object and the resolution of the display. Other workers* have found that, with their methods for the hidden surface problem, the computing time grew very rapidly with complexity. Thus, the display of significant objects was impractical.

There were other disadvantages. Roberts used rectangular solids and prisms to construct objects. This is a severe limitation when dealing with curved or Riemannian surfaces. To get around this difficulty, we have used triangles to describe objects. For example, it is easily seen that it is impossible to completely cover the surface of a sphere with quadrangles. However, it can be done quite conveniently with triangles (Figure 1).

---

* Lawrence G. Roberts
Lincoln Laboratory
Massachusetts Institute of Technology

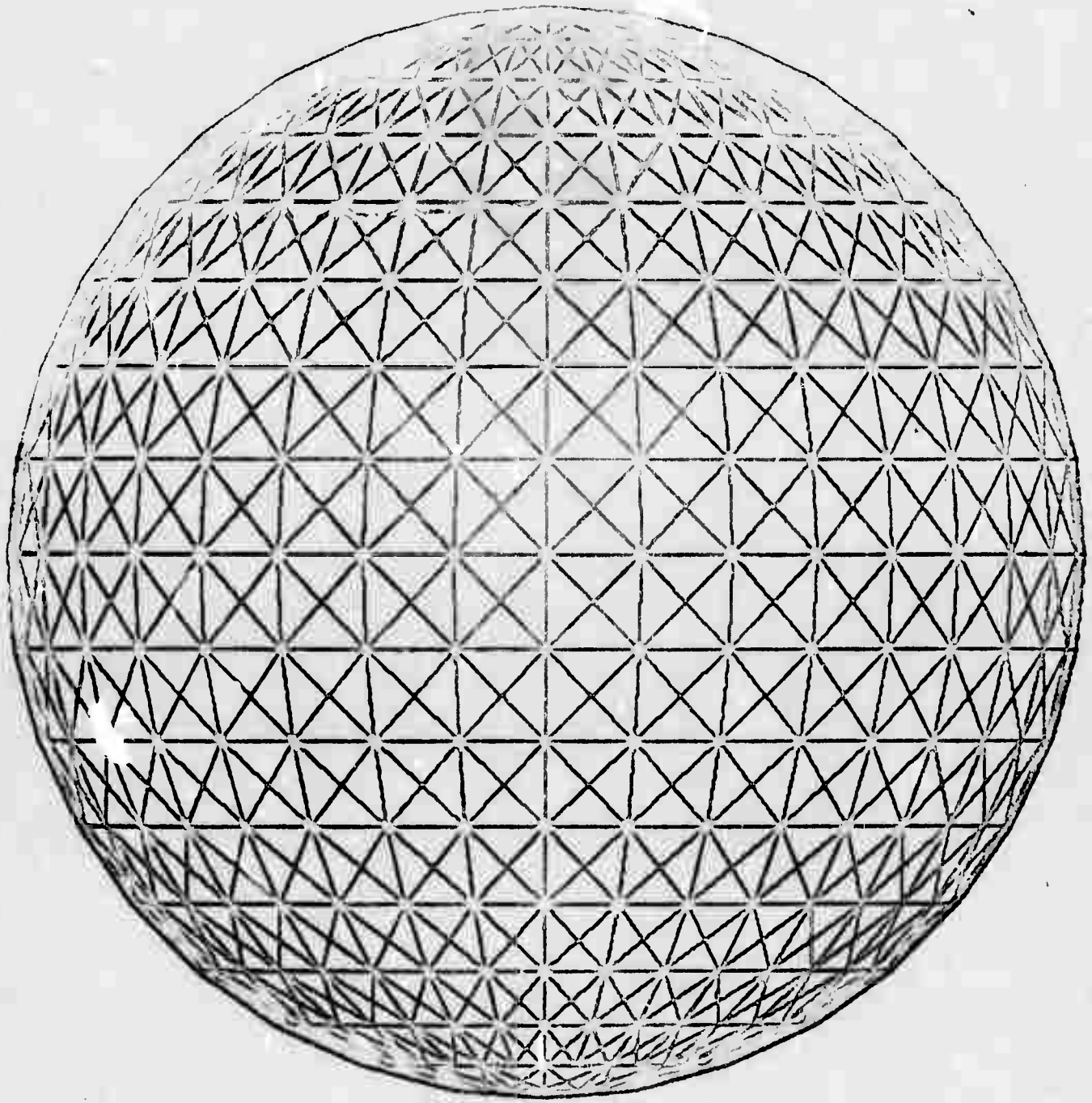


Figure 1. Example of one method of approximating a sphere by planar triangles.

Any developable surface can be approximated arbitrarily accurately with small, but finite, triangles (Figure 1). Another reason for using triangles is that three points always determine a plane. In this case many results from geometry and linear algebra have attractive forms for computation.

The object, and its perspective projection on a view plane, are examined by a scanning ray extending from a view point (Figure 2).

OVERVIEW OF THE ALGORITHM

I. Background Concepts

A. Basic geometry of the problem.

Everything is ultimately referred to an underlying, orthonormal vector basis $\vec{E}_j$ (Figure 3). The object is viewed from an arbitrary vantage point specified by $\vec{P}$. The viewing plane is parallel to vectors $\vec{e}_1$ and $\vec{e}^2$. The angular orientation of the viewer about the $\vec{e}^3$ axis is determined by $\vec{e}$ and $\vec{e}^2$. Every three-dimensional triangle determines a two-dimensional perspective image on the view plane. In Figure 2, triangles II and III are in front of triangle I.

B. Illumination of the object.

The present algorithm allows only a point source of illumination at the view point (like a single flashbulb photograph). As a consequence, there will be no shadows in the picture. The apparent brightness of a point on a surface depends on the following:

1. The basic physical laws governing incident light, e.g. light flux varies as the inverse square of the distance from a point source, and the amount of light incidence on the surface is a function of the angle of incidence (the angle between the incident ray and the normal to the surface).

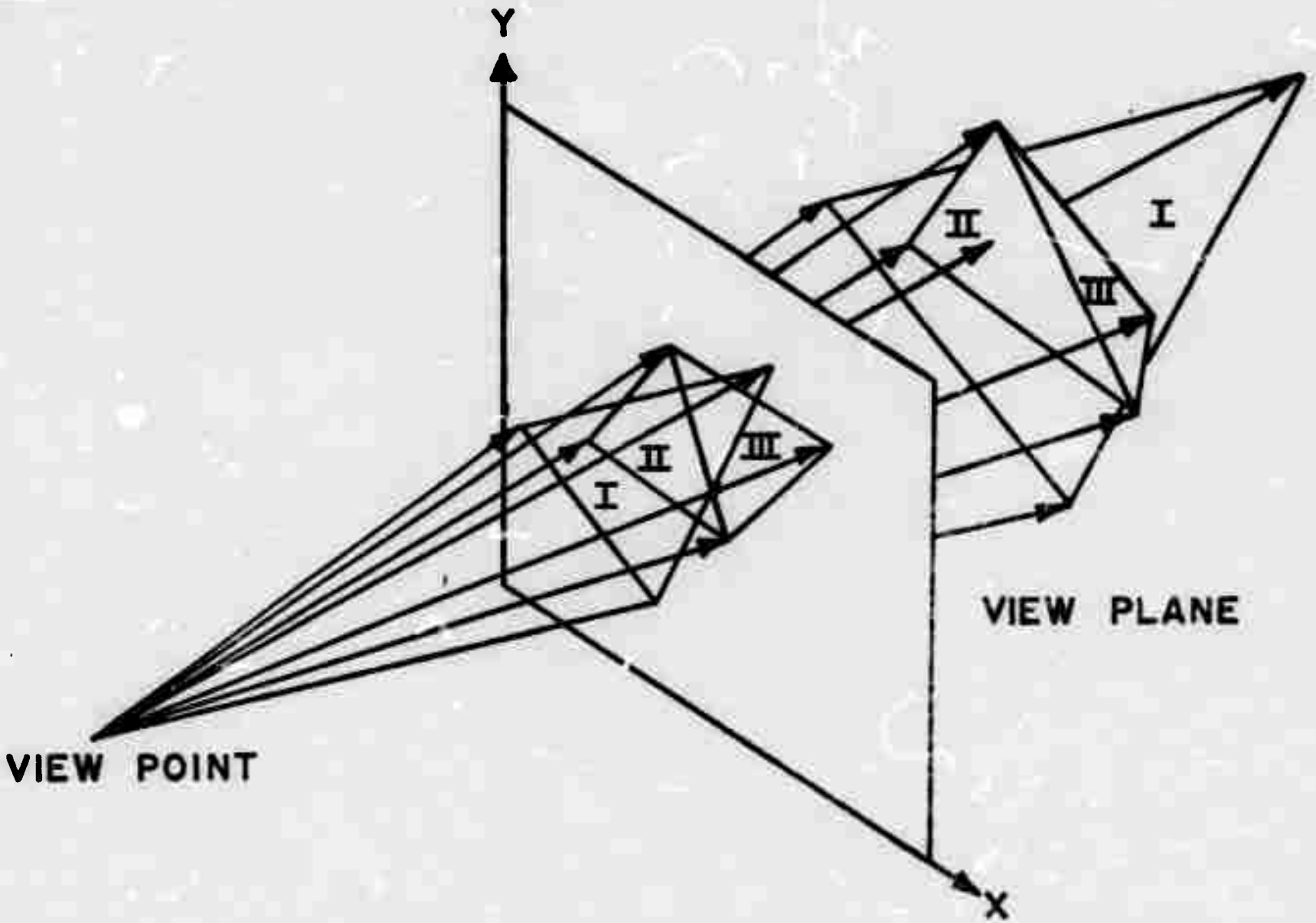


Figure 2. Perspective projection of triangles onto the view plane.

2. The nature of the reflecting surface, e.g., the reflectivity, texture and color may vary.

We have arbitrarily chosen an inverse fourth law for computational simplicity. The user, however, may employ any relationship he desires, by modifying the appropriate subroutine.

C. Hidden surfaces problem.

By far the major obstacle is solving the hidden surface problem and the means of preventing the computing time from growing faster than the number of triangles. Most of this paper will be devoted to this problem.

In solving the hidden surface problem, one could compare all the components of the entire surface for each point in the picture. This leads to a computation which likely grows at least as the product of the resolution and the number of surface elements. Instead, by using special sorting algorithms, only those triangles intersected by the scanning ray (Figure 2) need be compared.

II. Algorithm

This is a greatly simplified version to avoid getting bogged down in programming details.

A. We have already assumed that any object may be approximated by a set of triangles. The input data is a set of arbitrarily ordered triangles specified by the three-dimensional coordinates of their vertices (Figure 3).

The following must be specified (Figure 3):

1. View point, $\vec{P}$.
2. View basis, $\vec{e}_2$.
3. Distance from the view point to the view plane.

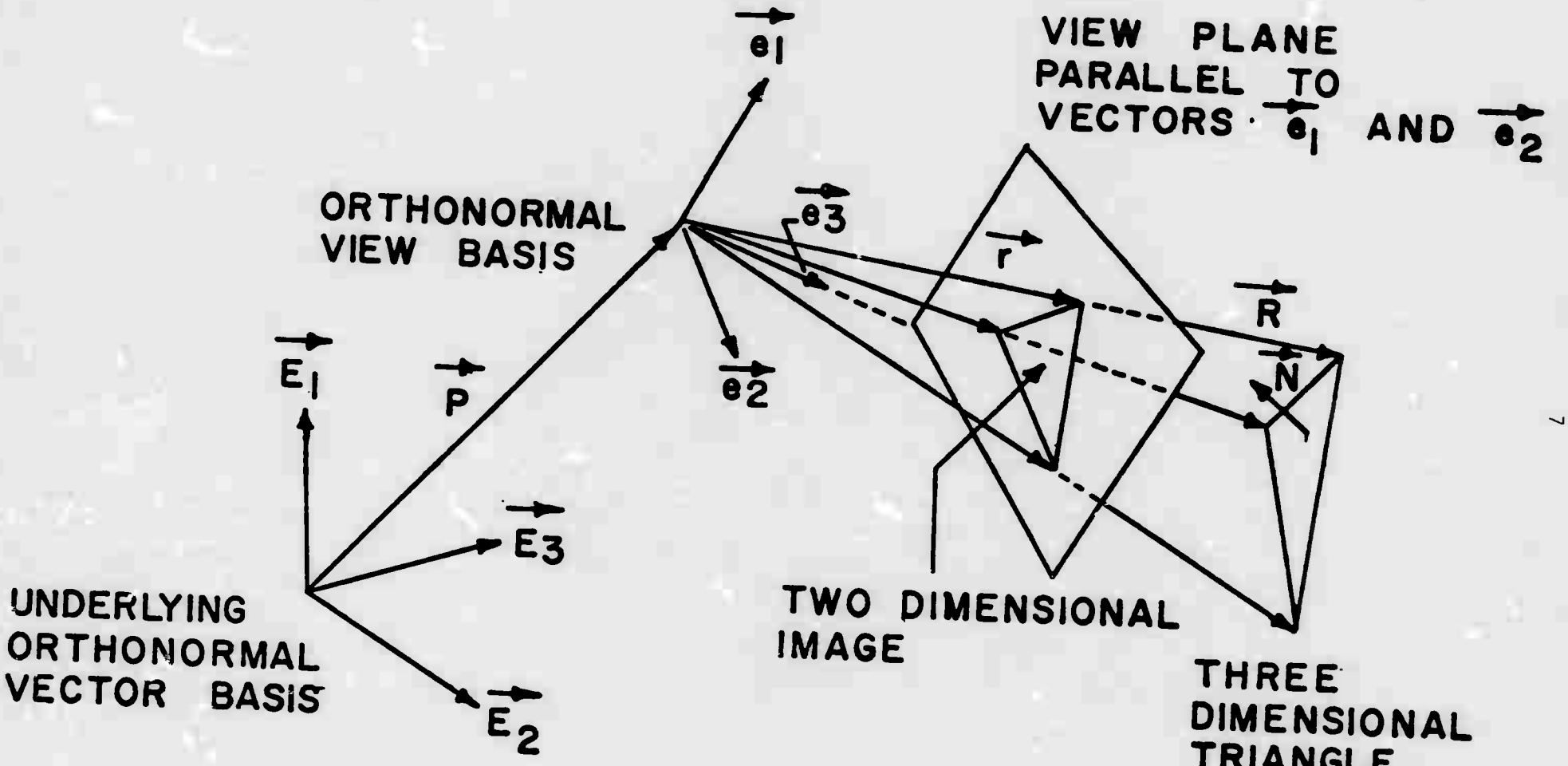


Figure 3. Basic geometrical convention for the algorithm. The unit normal to a triangle is $\vec{N}$.

B. Per frame calculations.

The view plane is examined by a systematic scanning raster (Figure 5). One complete raster scan of the view plane will be called a frame.*

1. Preprocess the triangles.

The algorithm begins with the calculation of various quantities about each triangle that will be needed for later computations. They are not, however, essential in order to understand the basic method.

a. Find the normal $\vec{N}$, to each object triangle.

This is needed in the apparent brightness calculation (Figure 3). From Figure 3, it is seen that the unit normal $\vec{N}$, to the three-space triangle may be calculated by normalizing the vector cross product of any two of the triangle's sides.

b. Find the apparent brightness at each of object triangle's vertices, assuming a point source of illumination at the view point.

1) To find these brightnesses, the distances from the view (or illumination) point to each of the three vertices of the triangle must be determined.

It is assumed that we have either been given (or have transformed) the components of all three-space position vectors so they are relative to the view basis.

*One side benefit of the raster scan is the inherent compatibility of the method with television-type display devices.

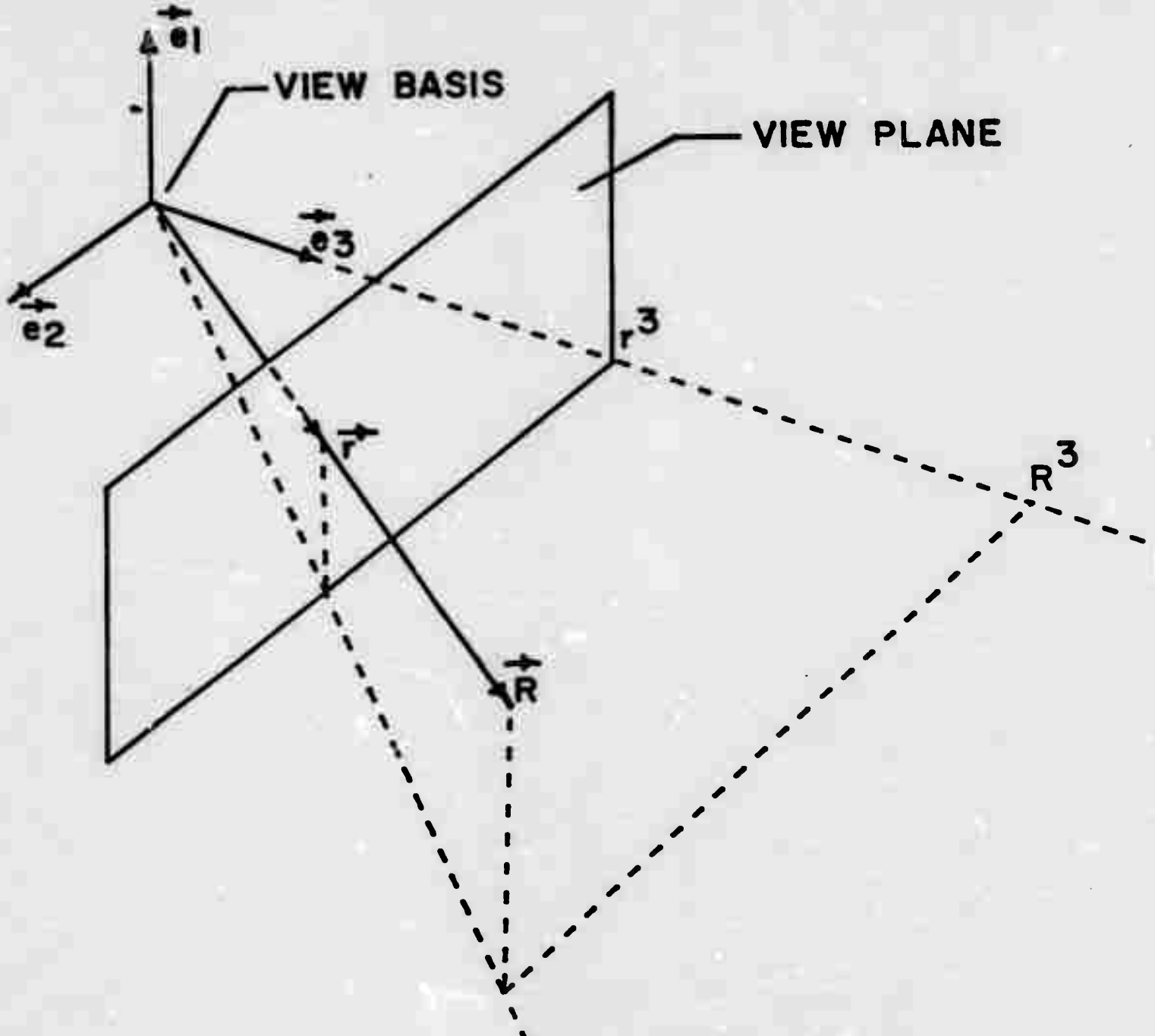


Figure 4. Example of the relationship of the components of $\vec{R}$ and $\vec{r}$.

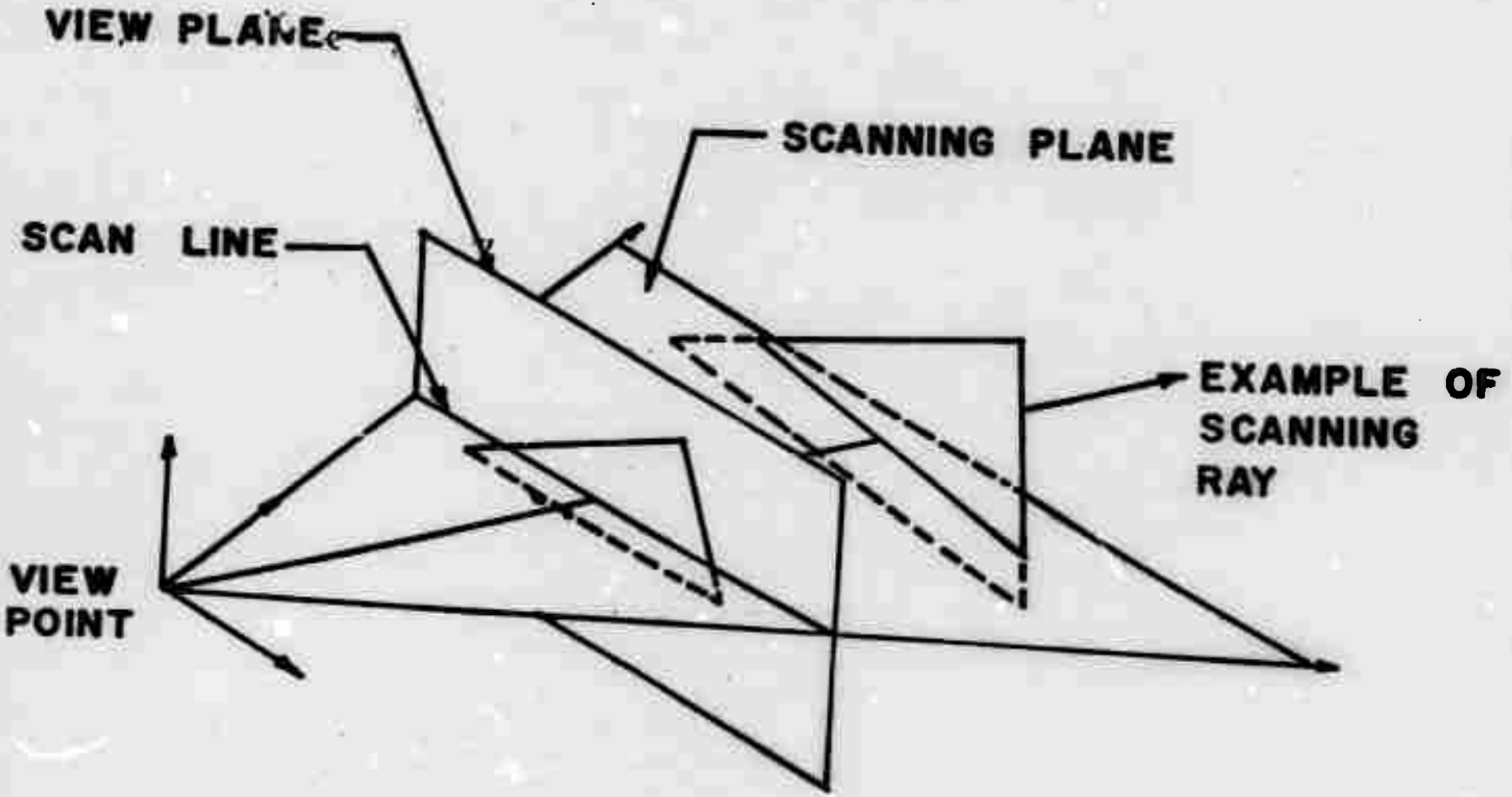


Figure 5. Slice made by the sweep of a scanning ray.

2) The magnitude of a position vector to a vertex is:

$$|\vec{R}| = \sqrt{(R_1)^2 + (R_2)^2 + (R_3)^2}$$

Given the unit normal of each triangle and the distance to each vertex, the user may calculate the apparent illumination of the triangle vertices with any formula he desires. The formula chosen generally will vary primarily with the nature of the reflective surface.

c. Calculate the linear brightness interpolation parameters as in Section II, E, 2.

Essentially, we have assumed that the apparent brightness in the interior of the view plane image of a triangle is adequately approximated by linear interpolation of the brightnesses at the three vertices of the triangle. The formula is easily derivable and will not be discussed here.

d. Calculate the linear distance ratio parameters.*

1) The distance ration w along any particular scan ray is defined to be:

$$w = \frac{\text{(Distance from view point to object)}}{\text{(Distance from view point to image)}}$$

In Figure 3, w = R/r for the scan ray $\vec{R}$ to the uppermost vertex. For a given view point, the equation of the plane

*This section is a correction of the method used in calculating the distance ratio parameters as detailed in AFIPS FJCC Proc. 31 p. 49, Nov. 1967. This error was discovered by Gordon Romney and is the basis of the algorithm presented in ARPA REPORT 4.3 and used here in order to clarify the AFIPS Proceedings.

of each triangle is found and its coefficients retained.* The plane is then

$$1 = ax^1 + bx^2 + cx^3$$

where x^1, x^2, and x^3 are view basis coordinates. From the definition of w, we see that

$$x^1 = wy \qquad x^2 = wx \qquad x^3 = wz$$

where x and y are the view basis coordinates of points on the view plane and z is the distance from the view point to the view plane and is constant throughout a frame. Using these relations we may rewrite the equation for the triangle plane in terms of view plane coordinates x, y, z.

$$1 = awx + bwy + cwz$$

Solving for $\frac{1}{w}$, we get

$$\frac{1}{w} = ax + by + cz$$

The quantities a, b and xz for each triangle are constants throughout subsequent calculations.

2. Later in the algorithm, it will be necessary to find $\frac{1}{w}$ for each triangle intersected by a given scan ray. By comparing the $\frac{1}{w}$'s for each of the triangles intersected, the triangle closest to the view point is easily found. Since z is constant, a scan ray is defined by the view plane coordinates x and y. Now x and y may be inserted directly into the above equation to find $\frac{1}{w}$ when needed.

---

*These are recomputed when and only when the view basis or, equivalently, the orientation, of the object is changed.

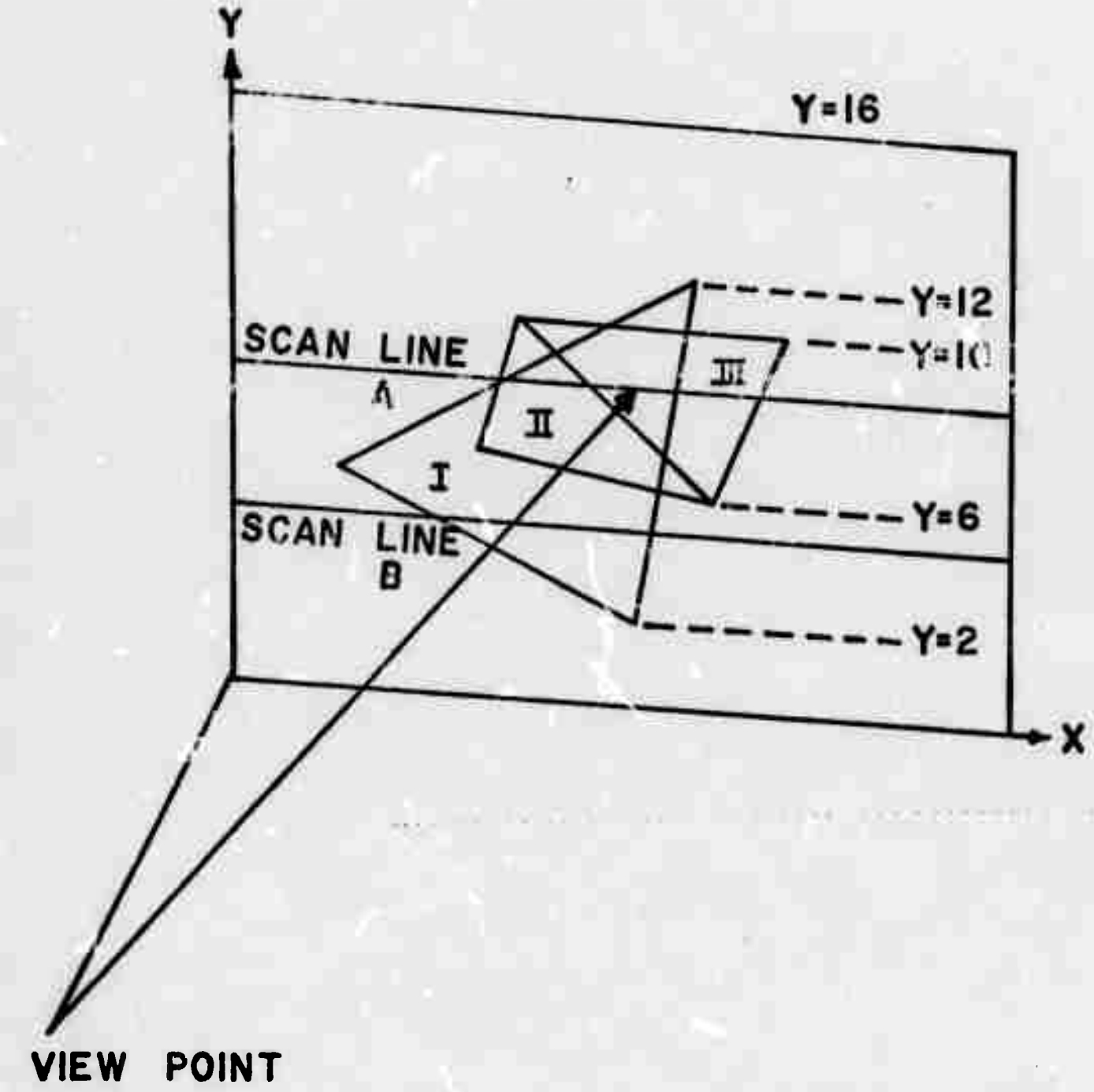


Figure 6. Scan line intersections of projected triangles on a simplified view plane.

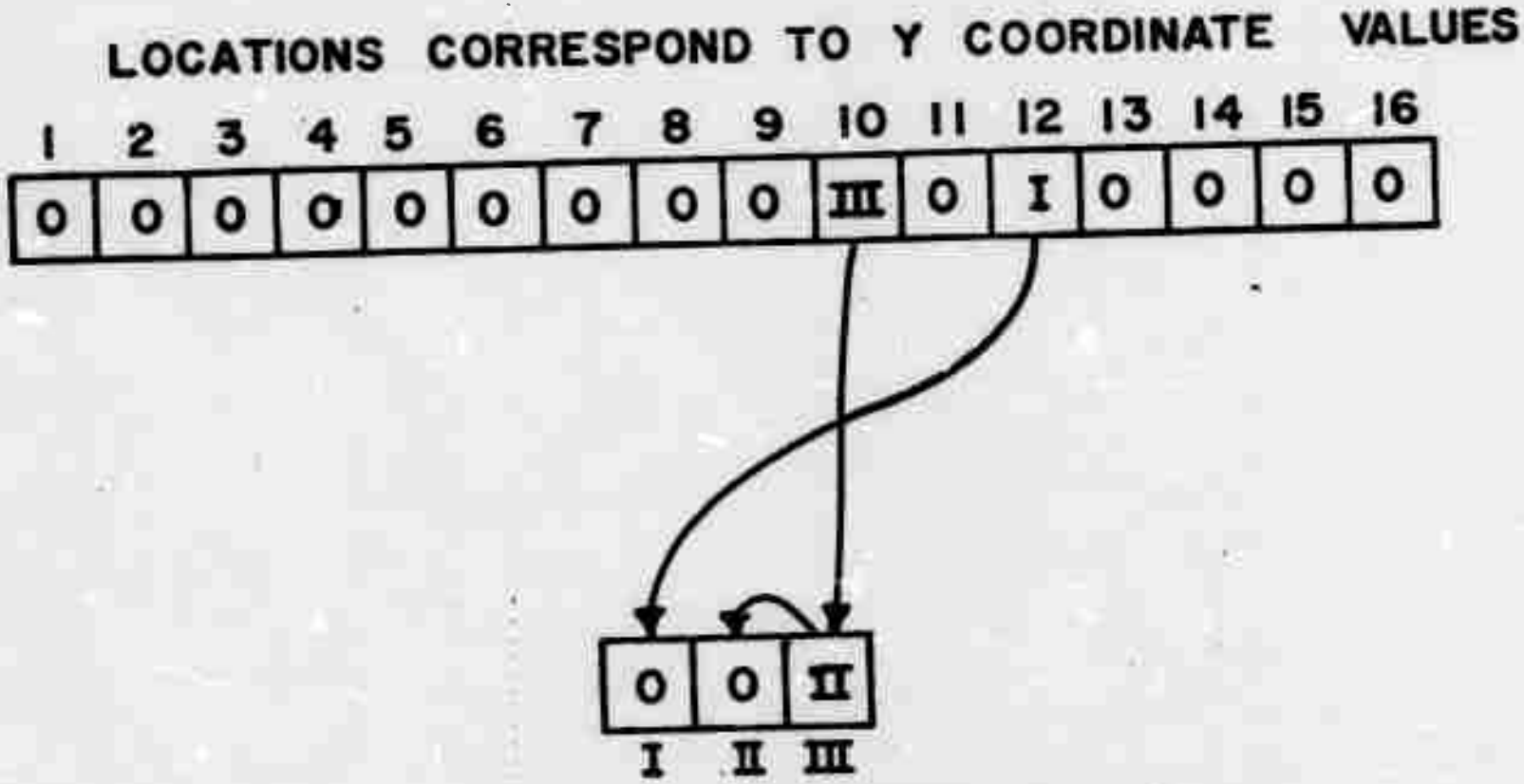


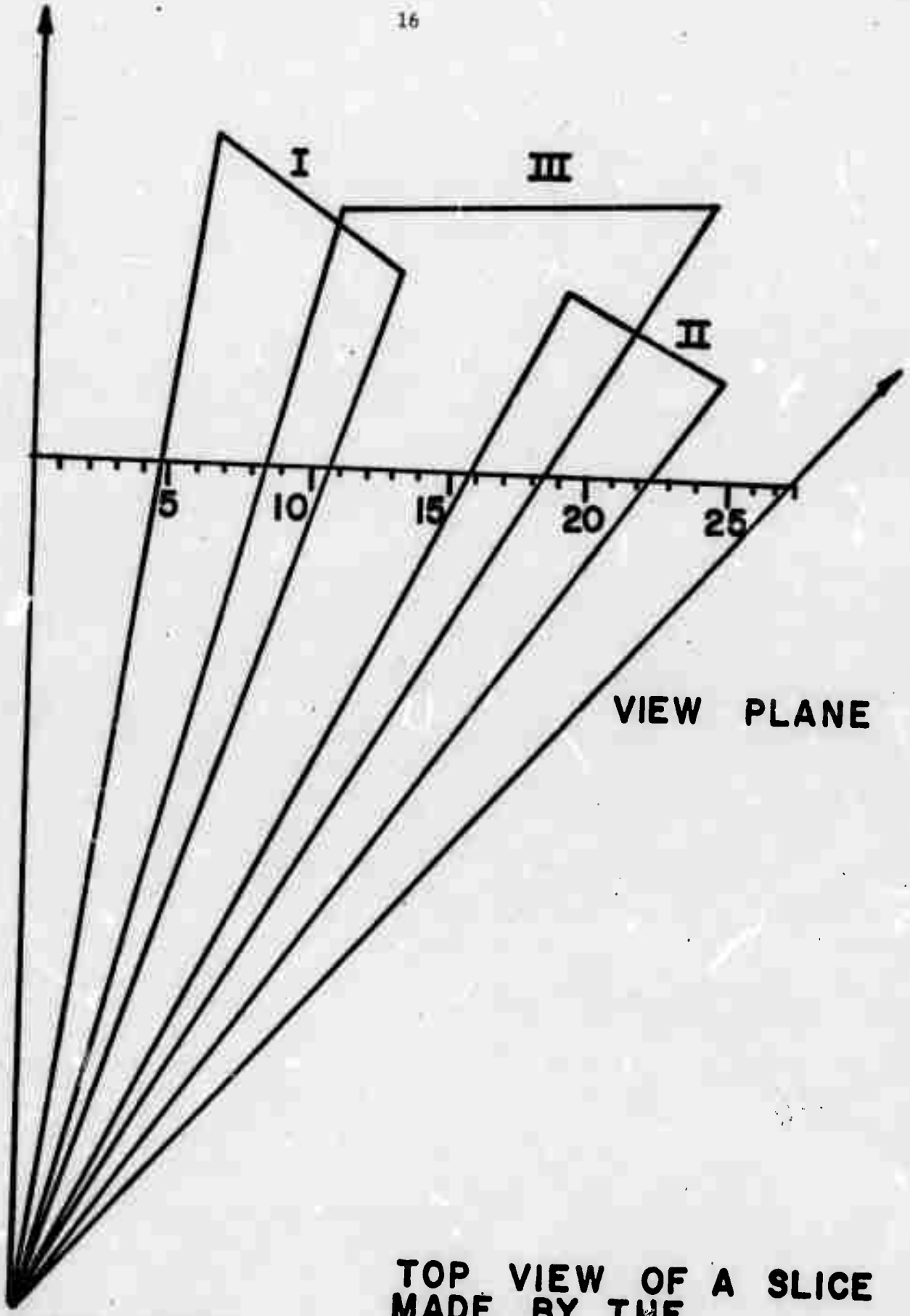
Figure 7. The y-entry tables corresponding to the triangles in Figure 6. This is essentially a pointer sort indicating which triangles enter at a given y.

TRIANGLE NUMBER →

I	II	III	
0	0	0	16
0	0	0	15
0	0	0	14
0	0	0	13
1	0	0	12
1	0	0	11
1	1	1	10
1	1	1	9
1	1	1	8
1	1	1	7
1	1	1	6
1	0	0	5
1	0	0	4
1	0	0	3
1	0	0	2
0	0	0	1

Y

Figure 8. For a given y scan, an occupied table indicates those triangles the scan line intersects. This figure shows a sequence of occupied tables for the entire segment of the view plane illustrated in Figure 6. Only a single row of the occupied table will exist at any instant for the current y.



TOP VIEW OF A SLICE MADE BY THE SCANNING RAY.

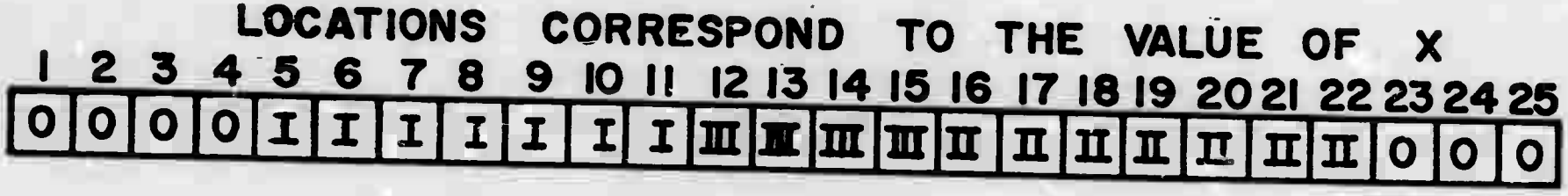


Figure 10. The visible table contains the triangle number visible at the designated x position of the scan ray.

ject all three-dimensional triangles onto the view plane to
of two-dimensional triangles (Figures 2 and 3).
e pro d image is going to be scanned from top to
bo a line at time, all of the triangles are sorted with
o y to e minate from consideration those triangles not
d by the rrent scan line (Figure 6).

a. Sort the three vertices of each triangle with respect to y.* This allows the convenient segregation of all of the entrance vertices from the set of exit vertices. The intermediate vertices are ignored.

b. The y-entrance table (Figure 7), tells which, if any, triangles are entered (i.e., begi o be intersected) by the scan line at a given y. It is built by sorting the set of y-entrance vertices.

c. The y-exit table, which is identical in structure to Figure 7, tells which triangles are exited by the scan line at a given y.

d. This way we will only have to look at those triangles that a given scan line actually crosses. This avoids examining all of the triangles in the entire picture at each scan line (Figure 6).

4. Start the y-scan.

Every time the scan is incremented by Δy, the y-occupied table (Figure 8) is up-dated, if necessary, by looking at the y-entry and exit tables to see if a triangle has been entered or exited.

---

*This sort actually switches the vertex data in core storage.

In Figure 6, the scan line moves downward and triangles are successively entered and exited. We turn on an occupied flag when a triangle is entered and turn it off when it is exited. Each triangle has its own location in which to put an occupied flag (Figure 8). Note that in Figure 8 only one row (or y-occupied table) of the sequence exists at a time. With a large number of triangles, the occupied table should be organized as some data structure allowing an efficient search of the table for occupied triangles. This is the first major reduction in computation.

C. Per scan line computations.

1. For the current scan line:

   a. We go to the y-occupied table and get only the triangles that this scan line crosses.

   b. Find the x values of the intersections of the scan line with the sides of the view plane images of the triangles pulled out of the y-occupied table (Figures 6 and 9).

2. Sort each triangle's x-entry and x-exit intercepts into the x-entry and x-exit tables, respectively. The x tables and sorts are identical to those used in the y-entry and y-exit sorts shown in Figure 7. They will not be shown in a figure.

3. Commence moving the scan ray along the scan line by Δx increments.

D. Per point calculations.

1. For the current x, look at the x-entry and x-exit tables to see if there is an intersection at this x. When an intersection point is encountered, we want to know if there is a change in the visible (or hidden) status of a triangle.

a. If there is an intersection

1) Update the x-occupied table. The x-occupied table has the same structure and is built in the same way as the y-occupied table.

2) Now go into the hidden parts calculation. (Section II, D, 2.)*

b. If there is no intersection, increment x and test for an intersection at the new x (i.e., increment x and go to Section II, D, 1).

2. Hidden parts calculation.

a. Examine all triangles in the x-occupied table for the current x.

b. Using the distance ratio parameters computed previously in per frame calculations (Section II, B, 1, d), calculate the distance along the scan ray from the view point to each of the three-space triangles entered in the x-occupied table (Figure 9).

c. Sort the distances to find the smallest distance. The triangle with the smallest distance ratio is the one visible.

3. Add to the visible table.

a. Figure 10 shows a completed visible table for the scan line shown in Figure 9.

b. The contents of the table are triangle names (or numbers) showing which triangle is visible at a given x.

---

*We assume that no triangles cross through any other triangles. In this case, the only time a triangle can change its visible or hidden status is at an intersection of the scan line and a triangle side (Figure 9). Intersecting triangles may be resolved into non-intersecting triangles by a separate algorithm prior to execution of the present half-tone perspective algorithm.

c. For each Δx increment of the scan ray, the visible table is modified by placing the name of the visible triangle, if any, in the location corresponding to the present x value (Figure 10).

d. The visible table is constructed for an entire scan line and is then used to find which triangle's intensity interpolation parameters are to be used for each x.

E. Per point intensity calculations.

Calculate the intensities for each x in the current scan line.

1. We interpolate to find the irtensity over the visible interior of a triangle using only the intensity values at the three vertices. This allows a considerable reduction in computing time. For simplicity and speed, but not necessity, we chose linear interpolation. The linear interpolation parameters have already been calculated and stored during the per frame calculations (Section II,B, 1, c).

2. The formula for the intensity at a point x on the scan line y is

$$I = ax + by + c$$

where I is the intensity and a, b, and c are the linear interpolation parameters for the visible triangle.

F. Output to display device

The list of intensities for this scan line is sent to a peripheral device for eventual display. The output subroutines are distinct and independent of the half-tone algorithm to permit flexibility as the display hardware is improved or altered.

G. In programming this algorithm, it should be noted that various combinations of triangles present cases that require arithmetic of great precision; e.g., triangles with common edges or vertices; triangles that are coplanar, or nearly so. These cases are not discussed here, but should be carefully examined by the programmer. From our experience, we have concluded that the description of the object and all computation should be done in integer arithmetic. Also, there should be a total avoidance of division to eliminate truncation error, which would make the logic of special cases ambiguous.

RESULTS

I. Program

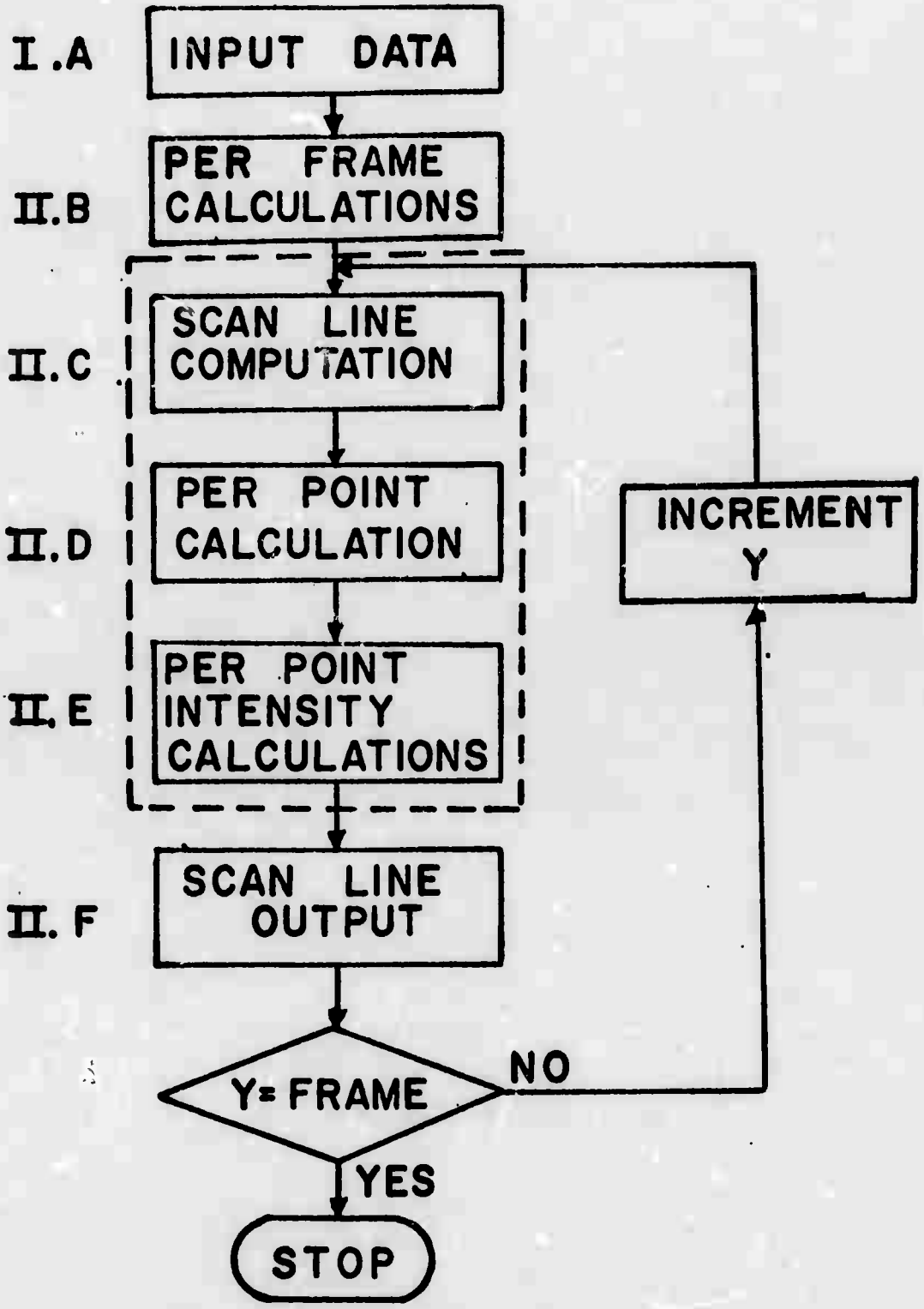
A FORTRAN IV program of the algorithm (Figure 11), called PIXURE, has been written* and used to produce half-tone pictures of a cube and tetrahedron (Figures 13 through 18). For both the cube (12 triangles) and the tetrahedron (4 triangles) the execution time of PIXURE was roughly 25 seconds to calculate a frame of 512 x 512 points on a Univac 1108. PIXURE at present is approximately 3800 Univac 1108 assembly language instructions in length and occupies 14K 36-bit words of storage for a picture of 100 triangle complexity.

Preliminary tests indicate that the execution time is most dependent on the number of scan lines that intersect the two-space image of the object (e.g., there are eleven scan lines, $2 \leq y \leq 12$, that intersect triangles in (Figure 6). It also appears that this dependence is closely linear. On the other hand, execution-time dependence on the number of triangles, i.e., the number of intersection points per scan line, appears to be much better than linear. The dependence on the number of hidden triangles per intersection point has not been rigorously determined, but seems to be close to linear.

---

*Dramatically improved versions and extensions of this algorithm have now been developed independently by G. Romney and J. Warnock. The resulting pictures are calculated much more rapidly and are of objects of some interest: e.g. combinations and intersections of cubes, tetrahedrons, planes and cones and a rather detailed house.

RELATED SECTIONS



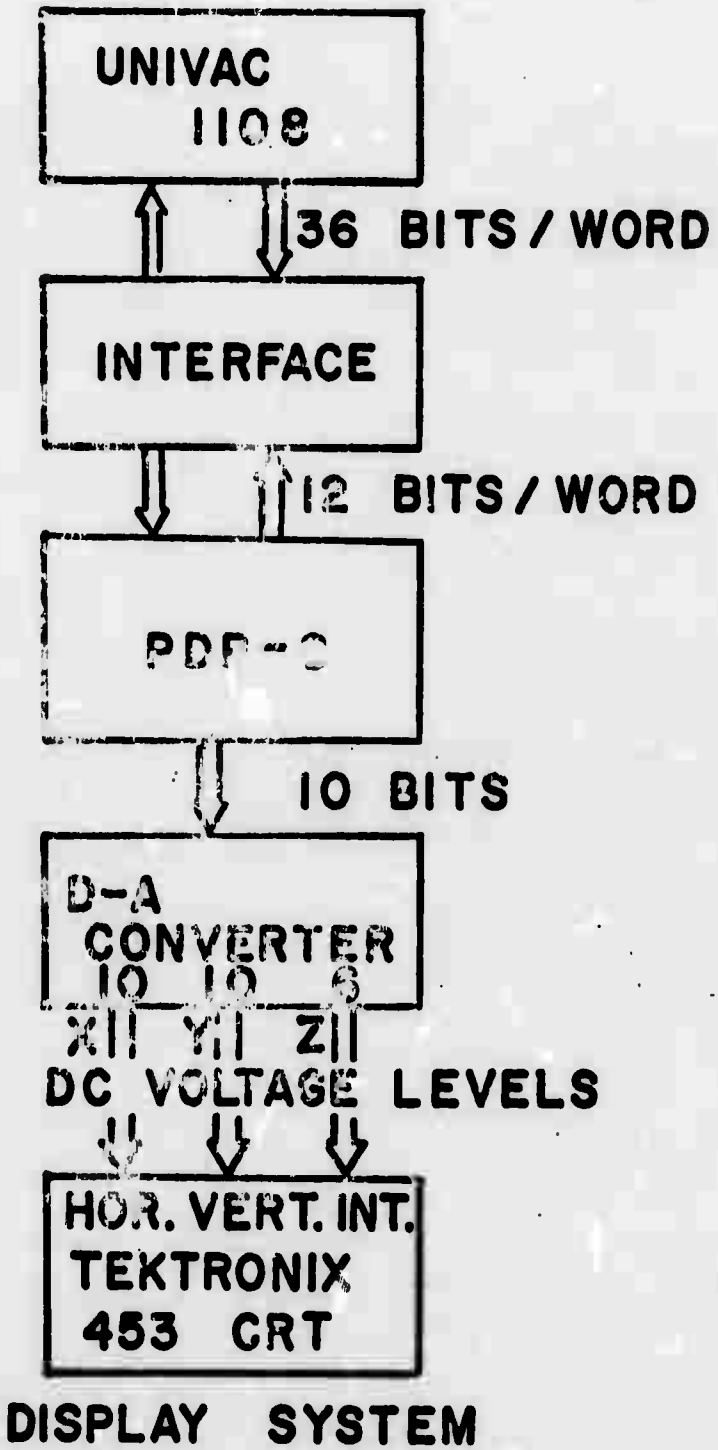


Figure 12. Display System.

II. Hardware techniques

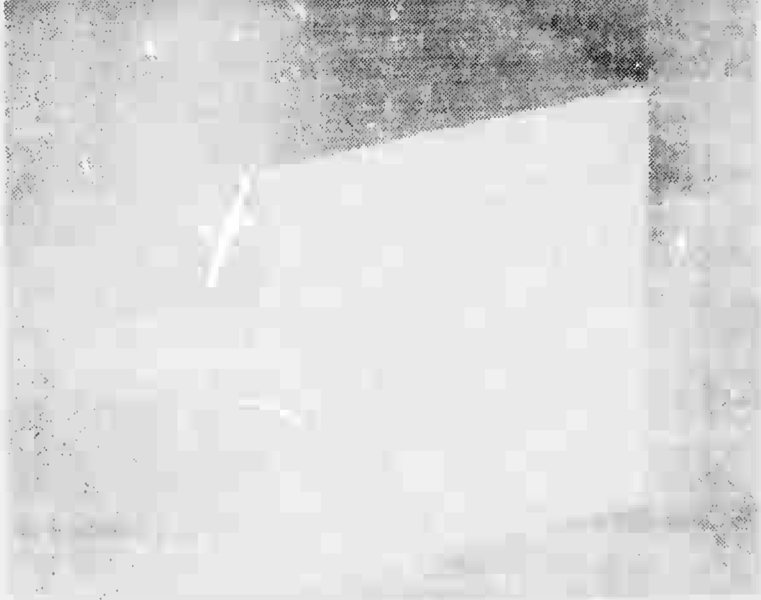
Each scan line that PIXURE generates is sent to a PDP-8 via a specially designed interface (Figure 12). The PDP-8 serves essentially as 1) a buffer, 2) a raster generating device for an oscilloscope, and 3) an a-synchronous I/O channel communicating with the 1108. Each scan line, in turn, is stored in the PDP-8 memory and then transmitted through a digital to analog (D-A) converter to a Tektronix 453 oscilloscope. The scan position is dictated by ten bit x and y registers in the D-A converter. The intensity of the beam at each point in the scan is controlled by a six bit z register. Due to storage and 1108-PDP-8 transmission-rate limitations, we have been forced to take time exposure photographs of the scope trace. As soon as a scan line is completed, the PDP-8 requests information for the next scan line. For a 512 x 512 frame it takes approximately ten seconds to generate a picture.

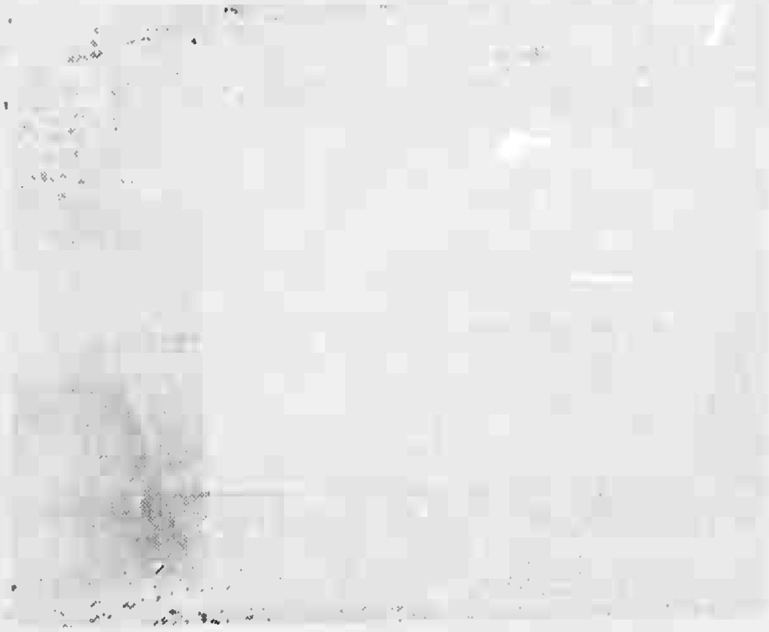
III. Subjective interpretations

The principal objective of this project is to allow people to see three-dimensional objects, as realistically as possible, using two-dimensional images (or displays). We have chosen to erase hidden surfaces and use half-tone shading to give the illusion of depth (or distance) and indicate spatial relationships. Although we are presently limited to a single source of illumination at the view point; nonetheless, the pictures of our test objects show obvious dimensionality.
Figures 13, 14, and 15 represent a cube whose resolution differs by a factor of ten. It is evident that Figure 14, representing a picture of 512 x 512 points, supplies sufficient information to adequately describe the cube. A more critical test on the resolution of the

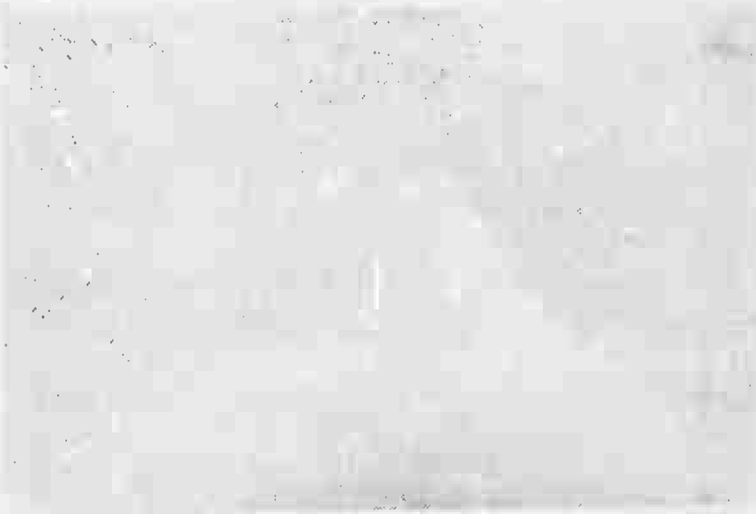
receding edge could not have been made, and yet, the edge appears in the higher resolution pictures. The unusual perspective, however, was merely the result of an arbitrary choice in geometry. The apparent triangular composition of the cube faces has since been corrected and a smooth transition across the triangle boundaries achieved (Figure 16).

The pictures of the tetrahedron (Figures 17 and 18) are superior in quality to those of the cube for two reasons. First, a defect in the display hardware was partially corrected, resulting in a more even display pattern. Scan lines are still noticeable, but it is felt that additional improvement in the hardware will significantly diminish this defect. The second improvement was in the selection of a more correct range of intensity levels used in the brightness calculation. Another objective is to display an object so that it will not be ambiguously interpreted. The tetrahedron in Figure 17 is decidedly convex, but Figure 18 could be either convex or concave unless the source of illumination is specified.





[illegible]



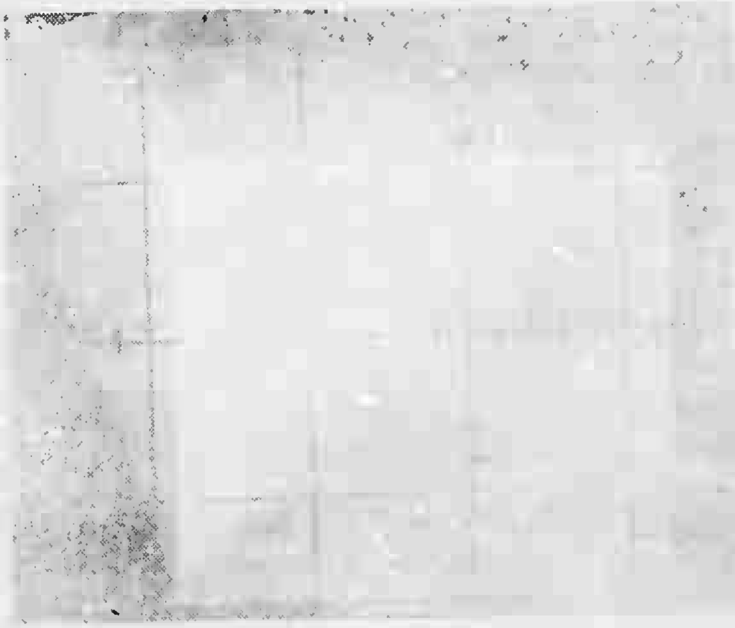
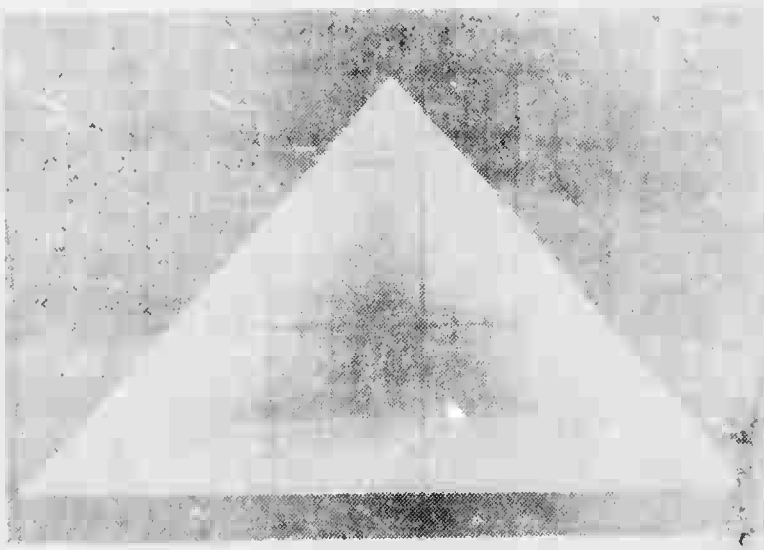


Figure 1[illegible]



[illegible] tetrahedr[illegible] 51[illegible]
(One side r[illegible]ved)

SUMMARY

In the cases we have tested, the computing time grows almost linearly with the resolution of the picture, the size of the visible portion of the object and, apparently, the amount of hidden surface. This makes the algorithm practical, and is a result of special sorting techniques which greatly reduce the number of hidden surface comparisons required. The objects we have displayed appear quite three-dimensional and their hidden surfaces are effectively eliminated. The computing time required for a picture composed of over 10^6 points was approximately 40 seconds on a Univac 1108.

The present system definitely proves the feasibility of the real-time display of two-dimensional half-tone images. It is felt that the technique may be easily extended to stereo representation of half-tone images. Furthermore, the algorithm is so constructed as to allow computations to be executed in parallel (see the dotted section in Figure 11). As many scan lines as hardware permits may be calculated simultaneously. Also, much of the computation may be performed by incremental hardware. The parallel and incremental characteristics of the algorithm lead us to believe that real-time movement and display of half-tone images is near realization.

A typical user wishes to describe an object in a form convenient for him. Also a flexible and extensive data structure must be constructed to contain and manipulate an object. Therefore, the practical application of the algorithm depends greatly on the ability of the system to convert an object into a suitable mesh of triangles. Our group has initiated work in these directions and, at present, has a triangle generation algorithm operational for objects composed of planar surfaces.

AKNOWLEDGMENTS

The authors are deeply indebted for the programming assistance of Lee Copeland and Richard Blackburn. The technical skills of Richard Jepperson, Charles Eder, and Y. T. Kim have assisted immensely in helping produce the first photographs. And, last but not least, we wish to thank the University of Utah Computer Center for their patience and assistance in making these results possible.

BIBLIOGRAPHY

Amir-Moez, A.R. and Fass, A.L., Elements of Linear Spaces (MacMillan Company, New York, 1962).

Jenks, F.G. and Brown, D.A., Three-Dimensional Map Construction, Science Vol. 154, No. 3750, 18 November 1966.

Meserve, B.E., Fundamental Concepts of Geometry, (Addison-Wesley, Reading, Mass., 1955).

Roberts, L.G., Homogeneous Matrix Representation of N-Dimensional Solids, (Lexington, Mass.: MIT Lincoln Laboratory).

Roberts, L.G., Machine Perception of Three-Dimensional Solids, (Lexington, Mass.: MIT Lincoln Laboratory, 1963), Technical Report No. 315.